**Remarks/Arguments**

## *Office Action Summary*

**Status.**

1. This *RESPONSE B* is in answer to the Office communication mailed 05/18/2006.

2. The Office communication is final.

3. NA

**Disposition of Claims.**

4. Claims 1-3, 5-17 and 19-28 are pending in the application.

5. No Claims have been allowed.

6. Claims 1-3, 5-17 and 19-28 are stand rejected.

7. NA

8. NA

**Application Papers.**

9. NA

10. The drawings as previously amended have been accepted.

11. NA

**Priority under 35 U.S.C. § 119.**

12. NA

## DETAILED ACTION

1. Claims 1-3, 5-17, 19-28 have been presented for examination based on the amendment of *RESPONSE A* filed on the 10$^{TH}$ February 2006.

2. Claims 1, 5, 10, 11, 12, 15, 19, 24, 25 and 26 were amended in *RESPONSE A*.

3. Claims 4 and 18 were canceled in *RESPONSE A*.

4. The arguments presented by applicant in *RESPONSE A* were considered by the Examiner, but all claims remain rejected.


### *Response to Applicant's Remarks & Examiner's Withdrawals*

5. The withdrawal of the objection to the oath or declaration is noted.

6. The withdrawal of the objection to the drawings is noted.

7. The withdrawal of the rejection(s) under 35 USC § 112 is noted.

8. Applicant notes that the Examiner maintains "the claim interpretation, as TLB and indexing table being claimed are similar."

   8.1. Applicant maintains the traversal of the "claim interpretation" to the extent that such interpretation has a bearing on the allowability of the claims remaining in the application. The traversal as presented in *RESPONSE A* is reaffirmed and incorporated herein and the patentable differences between an index and a Translation Look-aside Buffer (TLB) are further described in this *RESPONSE B*.

9. The withdrawal of the rejection(s) under 35 USC § 101 is noted.

10. The withdrawal of the rejection(s) under 35 USC § 102 is noted.


### *Response to Applicant's Remarks for 35 U.S.C. § 103*

11. **Claims 1-3, 5, 7-13, 15-17, 19, 21-27 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith**

**'1982 hereafter).** This response by the Examiner is respectfully traversed for the following reasons.

11.1.  In support of the rejection the Examiner argues as follows:

> *Applicant has argued that No one skilled in the art would believe that a simple indexing table is interchangeable with a TLB. Examiner agrees with the applicant, however the argument is not if an indexing table (of applicant) can perform the function of the TLB, but if the TLB (of the prior art) can be used as indexing table. In this case TLB can perform the function of the indexing table, therefore applicant's allegations are unpersuasive. MPEP 21 31.02 states:*
>
> > *"A generic claim cannot be allowed to an applicant if the prior art discloses a species falling within the claimed genus."*
>
> *The species in that case will anticipate the genus.*
>
> *Further, applicant's arguments are not directed towards the claimed limitations. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., indexing table) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See In re Van Geuns, 988 F.2d 11 81, 26 USPQ2d 1057 (Fed. Cir.*
>
> *Although examiner agrees that TLB and indexing table "are not the same thing" and the TLB as agreed by the applicant is a more "complex structure", however it meets the limitations recited in the claim. Applicant's arguments have been fully considered but they are not persuasive.*

11.2.  The Examiner's argument, quoted in Section 11.1 above, is in error in a number of ways.

11.2.1.  The Examiner's argument is in error in arguing that Applicant does not positively recite "storing into ... an indexing table".

11.2.1.1.  The Examiner admits that Applicant's claimed invention includes an "indexing table". Specifically, as quoted in Section 11.1 above, the Examiner's admission appears, among other places in the Examiner's statement "*Applicant*

*has argued that No one skilled in the art would believe that a simple indexing table is interchangeable with a TLB. Examiner agrees with the applicant, however the argument is not if an indexing table (of applicant)...*"

11.2.1.2.  Notwithstanding the Examiner's admission that an indexing table is present in applicant's claimed invention, the Examiner argues as quoted in Section 11.1 above "*the features upon which applicant relies (i.e., indexing table) are not recited in the rejected claim(s)*".

11.2.1.3.  Of course, the Examiner's argument is in error. Each of the independent Claims 1, 11, 15 and 25 positively an "indexing table". Since Applicant has recited an "indexing table" in all independent claims, all claims dependent thereon and hence all claims pending in the application directly or indirectly recite an "indexing table".

11.2.2. The Examiner's argument is also in error in arguing that a TLB is generic to Applicant's claimed "indexing table". In order to be generic, a TLB must generically be able to perform all of the functions of the species (allegedly applicant's "indexing table") and hence must be able to produce all of the advantages of the species (allegedly advantages of applicant's "indexing table"). Clearly, a TLB cannot meet such a standard for the reasons now described.

11.2.2.1.  The Examiner's states the following, in the quotation od Section 11.1 above, as being controlling "*... Examiner agrees with the applicant, however the argument is not if an indexing table (of applicant) can perform the function of the TLB, but if the TLB (of the prior art) can be used as indexing table. In this case TLB can perform the function of the indexing table,...*"

11.2.2.2. Clearly, the TLB of the prior art cannot perform the claimed indexing table and the claimed operation for the reasons now described.

11.2.2.3. An "indexing table" differs significantly from a "TLB" and that difference is recited in the claims, to wit, operations performed by an "indexing table" that are not performed by a TLB are recited in Claim 1 (with "A" and "B" added here for easy reference) as follows,

> (A) "if the indications indicate that said particular block has not been translated, going to said step of executing translated instructions,"
> (B) "if the indications indicate that said particular block has been translated, checking said translation store to determine if legacy instruction data has been modified"

11.2.2.4. In Claim 1, the "indexing table" goes directly to "executing translated instructions" in the "if ..." of (A) and does the checking in the "if ..." of (B). In a TLB, the checking must occur in the "if ..." of (A) and hence the efficiency of the present invention of maximizing performance for the "not translated" condition is missed by a TLB.
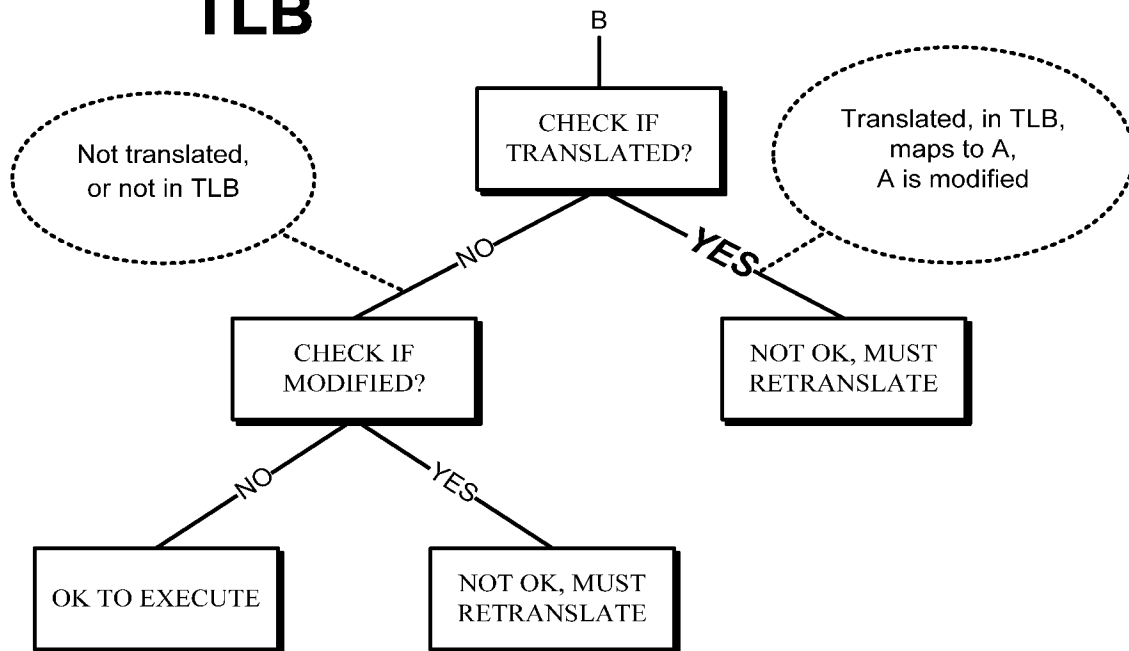
11.2.2.5. The differences discussed in Section 11.2.2.4 are seen in the following drawings which, when compared, depict the differences between the TLB and the INDEX TABLE. In the case of the TLB embodiment, the *YES* path is maximized since a direct one step transfer is made to "NOT OK MUST RETRANSLATE". However, such maximization of the **YES** path in a TLB **does not matter** since
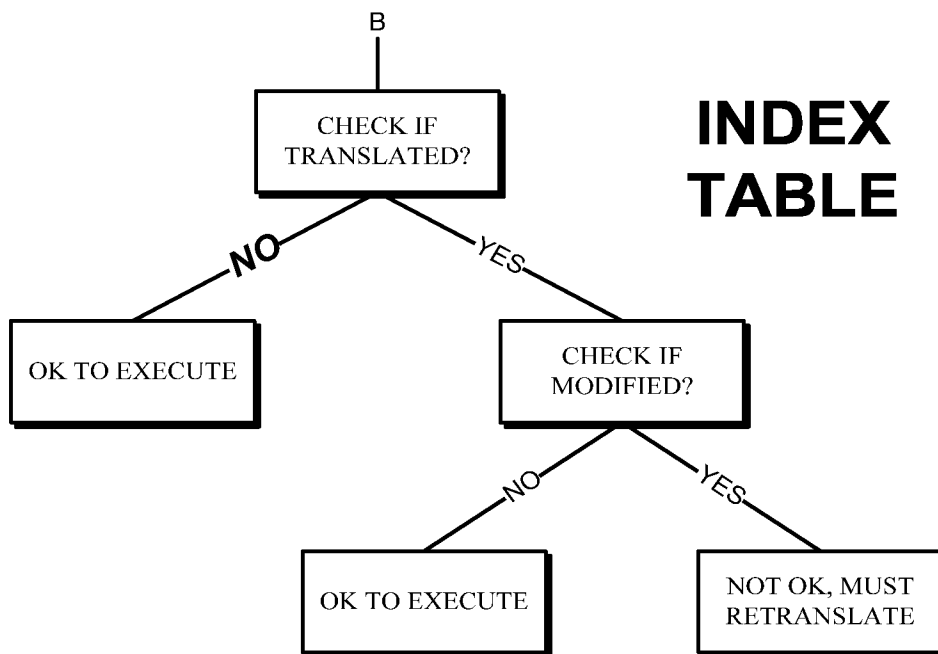
retranslation is required anyway. In the case of the INDEX TABLE embodiment, the *NO* path is maximized since only a direct one step transfer is necessary to the "OK TO EXECUTE". In the INDEX TABLE embodiment, such maximization **does matter** since retranslation is not required and it is "OK" to execute directly. Since this **NO** path that is maximized in Applicant's invention (See Claim 1, for example) using the INDEX TABLE statistically often occurs more frequently than the **YES** path, the maximization on the **NO** path is even more significant than might be expected.

11.2.2.6.  Nothing in any prior art cited by the Examiner has any suggestion of maximizing the **NO** path as is inherent in the A and B elements of Claim 1 as described in Section 11.2.2.3.

## TLB

B

CHECK IF TRANSLATED?

Not translated, or not in TLB

Translated, in TLB, maps to A, A is modified

NO

YES

CHECK IF MODIFIED?

NOT OK, MUST RETRANSLATE

NO

YES

OK TO EXECUTE

NOT OK, MUST RETRANSLATE

B

CHECK IF TRANSLATED?

## INDEX TABLE

NO

YES

OK TO EXECUTE

CHECK IF MODIFIED?

NO

YES

OK TO EXECUTE

NOT OK, MUST RETRANSLATE

11.2.2.7.   In summary, and in view of the clear differences in applicant's invention (using Claim 1 as an example), the Examiner's conclusions in the ***Response to Applicant's Remarks for 35 U.S.C. § 103*** cannot be maintained. Contrary to the Examiner's argument, a TLB is not generic since it cannot perform the claimed elements A and B of Claim 1 as analyzed in Section 11.2.2.3 above. Further, contrary to the Examiner's argument that the features relied upon by Applicant are not in the claims, an "indexing table" is recited for all claims in the application (See Sections 11.2.1.2 and 11.2.1.3).

11.2.3. The following argument of the Examiner, continuing with the quote of Section 11.1 above is also in error:

> *Further, in section 10.5 applicant argues that it is unclear how Mann'295 presents any problem that is solved by Smith '1982. Mann '295 teaches reuse of the translated code by multiple blocks (Mann'295: Col.6 Lines 62-Col.7 Line 3) which are chained together. The portion cited earlier is preceding paragraph citing the chaining mechanism. The reuse of the code by other block entry tables requires a many to one or one to many mapping between the translated code in the host code block and block entry tables. This deficiency is not addressed in Mann'295 and would be necessary for efficient code reuse (Problem to be solved). Smith '1982 teaching as described earlier in the rejection cures this deficiency. Hence the motivation to combine is well founded based on the implementation of the Mann'295 teachings.*

11.2.3.1.   The Examiner argues, as quoted in Section 11.2.3 above, "*The reuse of the code by other block entry tables requires a many to one or one to many mapping between the translated code in the host code block and block entry tables.*" Such operation is not stated or suggested in *Mann'295* and is not a *Problem to be solved* in *Mann'295*. *Mann'295*, as understood by Applicant, is a one-to-one operation. In *Mann'295* (see Col: 6. Line 62 to Col 7: Line 3), "the block entry tables 80 include links to block entry tables 80 for **other** Host code blocks 88". Nowhere can Applicant find that *Mann'295* operates with *reuse of the code by other block entry tables* as asserted by the Examiner. The reuse of code is by the same block entry table, not by other tables. The Examiner finds a *Problem to be solved* in *Mann'295* based upon an operation that is not described in *Mann'295*. Furthermore, the Examiner admits such operation is not described in *Mann'295*; in the Examiner's own words, "*This deficiency is not addressed in Mann'295...*". The reason it is not addressed in *Mann'295* is because *Mann'295* does not operate in the manner suggested by the Examiner and is not permitted to operate in the manner suggested by the Examiner. Since *Mann'295* does not operate in the manner suggested by the Examiner, the *Problem to be solved* suggested by the Examiner does not exist in *Mann'295*. Therefore, at least for this reason, there is no motivation to combine *Mann'295* with *Smith '1982*.

*11.2.3.2.*   For purposes of argument and without any admission intended, even if the Examiner's interpretation of the operation of *Mann'295* were correct (and it is not believed to be correct), still there is nothing in *Mann'295* that would provide one skilled in the art with motivation for combining *Mann'295* with *Smith '1982*. The Examiner has admitted that *"Applicant has argued that No one skilled in the art would believe that a simple indexing table is interchangeable with a TLB.*

*Examiner agrees with the applicant, ...* ". Similarly, no one skilled in the art would think that the complex structure of *Smith '1982* would be used as suggested by the Examiner to fix some unidentified problem (admitted by the Examiner as not being identified) in *Mann '295*. Furthermore, the operation suggested by the Examiner would add great expense to *Mann '295,* would slow down the speed of operation of *Mann '295,* and would add no benefit to *Mann '295.* Certainly, no one skilled in the art would adversely modify *Mann '295,* particularly when such modification achieves no benefit.

11.2.4. The following argument of the Examiner, continuing with the quote of Section 11.1 above, is also in error:

> *Further, applicant argues, outside the claimed limitations, that the teachings of Mann '295 and Smith'1982 even if combined to* (sic) *not teach or suggest applicant's invention as Mann'295 teaches re-translation in every case, whereas applicant's invention avoids re-translation providing savings in execution time. This is not correct, Mann teaches translation until a certain number of times, kept as a count, and once the count is reached, stores the translation (DOCT) in the host code block for reuse (Fig.4 Element 122, 126 and (Col.6 Lines 62-Col.7 Line 3) by other blocks. Further, Applicant's argue that such a combination (Mann '295 with Smith'1982) does not describe the avoidance of re-translation if instruction data has not been changed. Examiner respectfully disagrees, as Mann '295 teaches this limitation (Mann '295: Col.7 Lines 4-37) where new Host code block is assigned for re-translation if the instructions are modified.*

11.2.4.1. In the Examiner's argument quoted in Section 11.2.4 above, the Examiner is referring to Applicant's statements in Section 10.6 of the prior *RESPONSE A*. However, the Examiner's argument quoted in Section 11.2.4 above omits the key provision in Applicant's statements in Section 10.6 of the prior *RESPONSE A* that made it clear that Applicant was discussing the operation of *Mann '295* in the event of self-modifying code. The relevant point is that *Mann '295* operates when self-modifying code is reached to abort execution of translated instructions and transfers back to the "interpreter". The transfer back occurs in *Mann '295* regardless of whether the self-modifying code actually made a modification of data or not. This transfer back in *Mann '295* is very wasteful of processing time.

By way of contrast, the present invention performs a test to determine if in fact a modification has been made to data as a result of a self-modifying instruction and, if no such modification has been made, continues processing, without delay, the previously translated instructions.

11.2.4.2. In the Examiner's argument quoted in Section 11.2.4 above, the Examiner characterizes Applicant's argument as *"Further, Applicant's argue that such a combination (Mann '295 with Smith '1982) does not describe the avoidance of re-translation if instruction data has not been changed."* Next, the Examiner argues *"Examiner respectfully disagrees, as Mann '295 teaches this limitation (Mann '295: Col.7 Lines 4-37) where new Host code block is assigned for re-translation if the instructions are modified."* The Examiner entirely misses the point of Applicant's argument and mischaracterizes the operation of *Mann '295. Mann '295* upon reaching self-modifying code, transfers back to the interpreter regardless of anything else. *Mann '295* does not operate according to the claims in the present invention, that is, the present invention tests to determine if a modification has in fact been made as a result of processing self-modifying code and, if not, continues with execution of the translated instructions without a time-wasting retranslation and, of course, without time-wasting "interpretation". *Mann '295* upon detecting self-modifying code always transfers back to the interpreter (*Mann* Col 7:Lines 17-19; Col 7:Lines 24-27). Even further, when *Mann '295* transfers back, it is not for the purpose of retranslation, but is for the purpose of execution using the interpreter.

### *Claim Rejections - 35 USC § 103*

12. **Claims 1-3, 5, 7-13, 15-17,19, 21-27 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter).** This rejection is respectfully traversed for the following reasons.

12.1. Regarding Claim 1, the Examiner argues (subsection numbering added by Applicant for ease of reference),

12.1.1. *Mann '295 teaches a computer-implemented method of emulating execution of a legacy instruction (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29).*

12.1.2. *Further, Mann '295 teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). Mann '295 teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19).*

12.1.3. *Further, Mann '295 teaches storing translations into translation store as the host code block (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28) for each legacy instruction.*

12.1.4. *Further, Mann '295 teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).*

12.1.5. *Mann '295 teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).*

12.1.6. *Further, Mann '295 teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).*

12.1.7. *Further, Mann '295 teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).*

12.1.8. *Further, Mann '295 teaches if a legacy instruction is not a store instruction, going to step of executing translated instruction as performing the DOCT (Mann '295: Col.9 Lines 5-10; Fig. 4 Elements: 126, 124, Fig. 5 Element 134,136 & 148).*

12.1.9. *Mann '295 teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the (block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).*

12.1.10. *Mann '295 teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X - do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).*

12.1.11. *Mann '295 teaches checking if the instruction has been translated (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the legacy instruction data has been modified, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 -20). If the instruction*

*data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).*

12.1.12. ***On a closer review, Mann'295 also teaches using index/offset (partial addresses) from the instructions as index for the block entry table, thereby teaching the limitation of "said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for multiple different blocks" (Mann'295: Col. 6 Lines 16-24).***

12.1.13. *Mann'295 does not teach the details of the limitation presented above explicitly. Smith '1982 teaches that TLB having a hashing mechanism to map the virtual addresses (block numbers) to the real address (translated host code block) (Smith '1982: Pg.475 Col.2 Paragraphs 3-4). Hashing (done by taking an XOR or through randomized algorithm) depends on the number of bits selected, resulting in folding or overlapping (Smith '1 982: Pg.488, Col.1 Paragraph 1; Pg.489, Col.1). The hashing scheme selected by the applicant is extremely simplified version, as only one middle bit is selected to index the translation indication index table.*

12.2. The Examiner's argument as quoted in Section 12.1, with subsection numbering added for clarity, is not supported by *Mann'295* because the Examiner fails to distinguish between operations that are "interpretation" versus operations that are execution of "translated instructions". In order to understand the differences, a review of the operation of *Mann'295* is first warranted.

12.2.1. The general operation of *Mann'295* is depicted in FIG. 4. The commencement of operation is described in *Mann'295,* Col: 5 Line 64, where a Target instruction 76 is first interpreted by setting the code tag 72 to "C". Hence in the first iteration of FIG 4

after START will have the Tag="F", Tag="E", Tag="M", Tag="L" conditions are all "No" thus arriving at the Tag="C" test 106. Since in the first iteration the Tag has just been set to "C", the 106 test result is "Yes" and the counter is incremented in 114. Since the first operation is being described, the "Count>Max?" test of 122 result is "No" and hence the instruction is interpreted in 128. If it is determined that the instruction is self-modifying code, then special handling is required (see Col: 7, Lines 4-37). As a result of such special handling, the Target code 72 is set to "X" as described in FIG 5. Thereafter, the processing in FIG. 4 with Target code 72 set to "X", has all of the Tag="F", Tag="E", Tag="M", Tag="L", Tag="C" conditions "No" so that all processing reverts to the "Interpret Instruction" 128 mode of operation.

12.2.2. The specific operation of *Mann'295* treats self-modifying code in two ways. In the first way, the entire block containing the self-modifying instruction is marked with an "X" and hence all processing of that block is by "interpretation" through test 108 and interpretation in 108 in FIG 4 and not by execution of "translated instructions". In the second way, a first portion of the block (forming a "new first block") up to and not including the self-modifying instruction has the last entry marked with an "L" and processing is transferred to "interpretation" for self-modifying instructions. For a second portion of the block (forming a "new second block") after and not including the self-modifying instruction is marked with an "F" and is processed by executing "translated instructions" unless, of course, another self-modifying instruction is encountered in the "new second block" in which case the transfer to "interpretation" is again repeated. The only manner of processing self-modifying instructions in *Mann'295* is by "interpretation" and not by execution of "translated instructions".

12.2.3. By way of distinction in the present invention, each translated self-modifying instruction is processed by execution of "translated instructions". The execution of

| RespB_06-10-19_fi.doc | | 10/19/06-11:04 |
|---|---|---|
| Atty Doc No: AMDH-08152US0 | **Page 29 of 41** | **SC/Serial No. 09/992,130** |

"translated instructions" in the present invention occurs conditionally as a function of whether or not prior execution of such "translated instruction(s)" resulting from a self-modifying instruction has or has not actually resulted in a modification of data. If no modification of data has occurred in the present invention, the "translated instruction(s)" resulting from a self-modifying instruction are executed directly and efficiently without further processing (Distinguish *Mann'295* which always wastes time by reverting to interpretation for self-modifying instructions). If modification of data has occurred in the present invention, the "translated instruction(s)" resulting from a self-modifying instruction are retranslated and then executed directly. The differences between *Mann'295* and the present invention are several. *Mann'295* always reverts to "interpretation" when a self-modifying instruction is encountered and the present invention never need revert to "interpretation". *Mann'295* always has a time penalty in that upon encountering a self-modifying instruction, *Mann'295* always reverts to more time-expending "interpretation". In the present invention, a time penalty is experienced only under the condition that "translated instruction(s)" resulting from a self-modifying instruction has(have) actually resulted in a modification of data. The statistical frequency of occurrence of self-modifying instructions that **do** actually result in a modification of data in many applications is much lower than the statistical frequency of occurrence of self-modifying instructions that **do not** actually result in a modification of data. Accordingly, on a statistical basis, the present invention which does not require retranslation in the absence of a change in data is far superior to any operation in *Mann'295* that always reverts to time-wasting "interpretation".

12.2.4. Referring specifically to the Examiner's rejection of Claim 1 as quoted in Section 12.1 above, the Examiner's statement quoted in Section 12.1.4 is directed to the element "*storing translation indications, ...*" in Applicant's Claim 1. In making the rejection as to that claimed element, however, the Examiner ignores the part of that

claimed element that recites "*said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for multiple different blocks,*". The Examiner ignores this recitation of *multiple different blocks* in Applicant's Claim 1 because *Mann'295* is a one-to-one structure that neither requires nor permits *multiple different blocks*. The Examiner does not identify anything in *Mann'295* that corresponds to Applicant's recitation "*said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for multiple different blocks,*". At least for this reason, the Examiner's rejection cannot be sustained.

12.2.5. Referring specifically to the Examiner's rejection of Claim 1 as quoted in Section 12.1 above, the Examiner's statement in Section 12.1.10 "*going to the step of executing the translated instruction*" is erroneous in that no execution of a "translated instruction" occurs, but rather as is clear from the Examiner's citation *(Mann '295: Fig.4 Path 108, 128, 129),* processing reverts in *Mann '295* to a time-wasting "interpretation" in "Interpret Instruction 128" as shown and described in connection with FIG 4 of *Mann '295*. At least for this reason, the Examiner's rejection cannot be sustained.

12.2.6. Referring specifically to the Examiner's rejection as quoted in Section 12.1 above, the Examiner's statement in section 12.1.11, to the extent that the Examiner suggests that in *Mann '295* "translated instruction(s)" from a self-modifying instruction are retranslated and then executed is in error. Unlike the present invention, a self-modifying instruction in *Mann '295* always reverts to a time-wasting "interpretation" rather than more efficient processing to execute a "translated instruction". The Examiner ignores the fact that the processing of the "translated instructions" of the original block in *Mann '295* is carried out by partitioning the original block into two

smaller blocks. The partitioned two smaller blocks in *Mann '295* include a "new first block" including instructions **up to and not including the original self-modifying instruction** and a "new second block" including instructions **after** the "translated instruction(s)" for the self-modifying instruction. The processing in *Mann '295* of the self-modifying instruction is only done, contrary to the Examiner's argument, by a time-wasting "interpretation" and is not done by a retranslation and then execution of the retranslated instructions. At least for this reason, the Examiner's rejection cannot be sustained.

12.3.  In making the rejection of Claim 1, the Examiner further argues,

> *Mann'295 does not teach the details of the limitation presented above explicitly. Smith '1982 teaches that TLB having a hashing mechanism to map the virtual addresses (block numbers) to the real address (translated host code block) (Smith '1 982: Pg.475 Col.2 Paragraphs 3-4). Hashing (done by taking an XOR or through randomized algorithm) depends on the number of bits selected, resulting in folding or overlapping (Smith '1982: Pg.488, Col.1 Paragraph 1 ; Pg.489, Col.1). The hashing scheme selected by the applicant is extremely simplified version, as only one middle bit is selected to index the translation indication index table.*

> *It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Smith '1982 and apply them to Mann '295 to implement the indexing table as disclosed. The motivation would have been that Smith '1982 discloses the necessity for TLB like lookup when dealing with translated information when address space doesn't map directly (Smith '1982: Pg.510, Section 2.9, 2.17). Mann '295's design requires translation as one target code block may have one or more translated host code*

*instructions. Hence Smith '1982 solves Mann '295's problem of mapping the target legacy instruction object to translated host object code block (Mann '295: Col.6 Lines 47-55). Please see further the response to arguments for clarification.*

12.3.1.  The arguments of the Examiner as quoted in Section 12.3 above are formulated by the Examiner only after reading and finding a suggestion therefor from Applicant's specification. The Examiner has already admitted that one skilled in the art would not equate a simple indexing table with a TLB (See Section 11.2.2 and subsections there under which address this argument and which are incorporated by reference here). Furthermore, the indexing table of *Mann '295* requires a one-to-one indexing and changing to a many-to-one operation, as suggested by the Examiner, would destroy the simplicity, lower cost and intended one-to-one operation of *Mann '295*. Such modifications to *Mann '295* would render *Mann '295* inoperable because there are no facilities in *Mann '295* for processing in a many-to-one environment. The suggestion for modifying *Mann '295* to be and operate like Applicant's invention can only be found in Applicant's specification. Certainly, one skilled in that art would find no reason and certainly no suggestion for changing the simple and inexpensive one-to-one operation of *Mann '295* to a very complex TLB many-to-one operation that provides no benefit to *Mann '295*. The Examiner only finds such motivation from Applicant's specification, there are no legitimate technical reasons why any skilled one in the art would choose to modify *Mann '295* in the manner suggested by the Examiner.

12.3.2. Furthermore, even if *Mann '295* were modified to include a TLB as suggested by the Examiner, the operation would not be the same as the index table and operation claimed by Applicant (See Section 11.2.2 and subsections 11.2.2, 11.2.3, 11.2.4, 11.2.5 and 11.2.6 which are incorporated by reference here).

| RespB_06-10-19_fi.doc | | 10/19/06-11:04 |
|---|---|---|
| Atty Doc No: AMDH-08152US0 | **Page 33 of 41** | **SC/Serial No. 09/992,130** |

12.4.  The arguments above while using Claim 1 as representative, are applicable and presented for all of the Claims 1-3, 5, 7-13, 15-17,19, 21-27 and reconsideration thereof is respectfully requested.

12.5. Regarding Claim 2, the Examiner argues *"Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63)."* Applicant traverses this rejection as Applicant can find nothing in *(Mann '295: Col.5 Lines 54-63)* that supports the Examiner's argument. Quite to the contrary, *(Mann '295: Col.5 Lines 54-63)* indicates that information is stored for all translated blocks on a one-to-one basis.

12.6. Regarding Claim 5, the Examiner argues *"From the teachings of Mann'295 and Smith '1982 it obvious to use the subset (middle 3 hexadecimal) of digits form the legacy instruction for the part of the address field of the indexing table (Mann'295: Col. 6 Lines 16-24; Smith'1982: Pg.475 Col.2 Paragraphs 3-4; Pg.488, Col. 1 Paragraph 1; Pg.489, Col. 1)."* Applicant traverses this rejection as Applicant can find nothing in *Mann '295* that would permit a many-to-one operation. Furthermore, even if such operation were permitted, nothing in either reference suggests the simplicity of using only a single center hexadecimal digit for controlling operations.

12.7. Regarding Claim 10, the Examiner argues as follows (with Section numbers added by Applicant for easy reference):

12.7.1. *Mann '295 teaches storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).*

12.7.2. *Further, Mann '295 teaches that the translation indication includes a state field (tag field), for each block number, indicating the block has been modified (Mann '295: Col.5 Table I; Col.6 Lines 62-67). Further, Mann '295 teaches incrementing the state field each time block is executed on the cache (Mann '295: Col.6 Lines 64- 65).*

12.7.3. *It would have been obvious to keep track of modification to the block as modifications are already kept track at each legacy instruction level (Mann '295: Col.7 Lines 13-38). Mann '295 does not teaches decrementing the count, but teaches equivalent functionality of removing and de-allocating the block entry table (indexing table entry) when the translated host block is no longer needed/garbage collected (Mann '295: Col.7 Lines 62-67, Col.8 Lines 1-3). Further, Mann '295 teaches bypassing the checking if there is no store instruction data (Mann '295: Fig. 5 Element 13411 36 & 148).*

12.7.4. The rejection of Claim 10 as quoted in Section 12.7 above is respectfully traversed for the following reasons.

12.7.4.1. The first recited element of Claim 10 requires storing a *"subset of all of the translated blocks"* As discussed in connection with Claim 1, *Mann '295* does not process subsets (See Section 12.2.4) and hence the Examiner has cited no art that teaches the first recited element of Claim 10.

12.7.4.2. The second recited element of Claim 10 requires *"a state field storing a count for each block number indicating whether the block represented by said block number has been modified"*. The Examiner's citation to *(Mann '295: Col.6 Lines 62-67)* does not count each time a block is "modified", but rather counts each time

the block is "executed". Accordingly, the citation by the Examiner does not teach the second recited element in Applicant's Claim 10.

12.7.4.3.   Since *Mann '295,* per the discussion in Section 12.7.4.2, does not have the required *state field* for counting modifications, *Mann '295* cannot have the third (increment count), fourth (decrement count) and fifth (check when count is zero) recited elements of Claim 10. The Examiner's arguments with respect to Claim 10 are without merit for the reasons described.

12.8. Regarding Claim 11, the Examiner argues as follows (with Section numbers added by Applicant for easy reference):

12.8.1. *Mann '295 teaches a method of dynamic emulating execution of a legacy instruction (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29).*

12.8.2. *Further, Mann '295 teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). Mann '295 teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19).*

12.8.3. *Further, Mann '295 teaches storing translations into translated code into host code block (translation store) for each legacy instruction (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28).*

12.8.4. *Further, Mann '295 teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches*

*indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).*

12.8.5. *Mann '295 teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).*

12.8.6. *Further, Mann '295 teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).*

12.8.7. *Further, Mann '295 teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).*

12.8.8. *Further, Mann '295 teaches checking store instruction associated to the block entry table, if instruction data is stored (Mann '295: Col.9 Lines 5-1 0; Fig. 4 Elements: 126, 124, Fig. 5 Element 134, 136 & 148) for a particular block. Mann '295 also teaches checking if the instruction data has been modified (Mann '295: Col.9 Lines 9 - 20).*

12.8.9. *Further, Mann '295 teaches bypassing the checking if there is no store instruction data (Mann '295: Fig. 5 Element 134, 136 & 148). Further, Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63).*

12.8.10. *A. Mann '295 teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the*

*(block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).*

12.8.11.    *Mann '295 teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X - do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).*

12.8.12.    *Mann '295 teaches checking if the instruction has been translated (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the legacy instruction data has been modified, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 -20). If the instruction data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).*

12.8.13.    *Mann '295 teaches storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).*

12.8.14.    *Mann '295 teaches translation indication includes a state field as tag field for each block number indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67).*

12.9. The Examiner's argument with respect to the rejection of Claim 11, as quoted in Section 12.8 is traversed for the reasons presented in connection with Claim 1. In particular, the Sections 12.1.1; ...; 12.1.7 correspond to Sections 12.8.1; ...; 12.8.7, respectively, and Sections 12.1.9; 12.1.10 and 12.1.11 correspond to Sections 12.8.10; ...; 12.8.11 and 12.8.12, respectively. Any remarks directed to such corresponding Sections in Claim 1 are reasserted in

connection with corresponding respective ones of the Sections in Claim 11 and are incorporated by reference here.

12.10.    Additionally, the Examiner's statement in Section 12.8.8 that "*Mann '295 also teaches checking if the instruction data has been modified*" is not supported by the Examiner's citation therefor, "*(Mann '295: Col.9 Lines 9 -20)*", nor by anything else in *Mann '295*. Rather, *Mann '295* only checks to see if a store instruction is present for storing into the Target instruction but does not check to determine if data is actually modified. Whether data is actually modified is irrelevant since an "X" is immediately marked and *Mann '295* transfers to "interpretation". Such operation is distinguished from the present invention where, if data has not been modified for self-modifying instructions, the self-modifying instructions are executed. A transfer is made to retranslate in the present invention only if data has actually been modified unlike in *Mann '295* where a time-wasting transfer to "interpretation" is always made.

12.11.    Additionally, the Examiner's statement in Section 12.8.9 that "*Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks*" is not supported by the Examiner's citation therefor, "*(Mann '295: Col.5 Lines 54-63)*", nor by anything else in *Mann '295*. Rather, the citation *(Mann '295: Col.5 Lines 54-63)* relied upon by the Examiner indicates that not all blocks are translated, the primary operation being "interpretation". Only after blocks have been repeatedly interpreted a number of times are the blocks translated. Once blocks are translated in *Mann '295*, they are stored with a one-to-one indexing with no provisions for an subset operations. (See Section 12.2.4 and discussions of subsets in other Sections).

12.12.    The additional arguments of the Examiner directed to Claims 12, 13, 15, 16, 17, 19, 21 22, 23, 24, 25, 26 and 27 rely upon limitations and arguments for prior claims and

Applicant's arguments for such prior claims are incorporated here for Claims 12, 13, 15, 16, 17, 19, 21 22, 23, 24, 25, 26 and 27.

13. **Claims 6 & 20 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter) in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).** This rejection is respectfully traversed for the following reasons.

13.1. Regarding Claim 6 and Claim 20, the rejection relies on the rejection of Claim 1 and the present rejections are traversed at least for the reasons discussed above in connection with Claim 1.

14. **Claims 14 & 28 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter) in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter), further in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).**
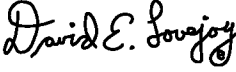
14.1. Regarding Claim 14, the rejection relies on the rejection of Claim 11 and the present rejection is traversed at least for the reasons discussed above in connection with Claim 11.

14.2. Regarding Claim 28, the rejection relies on the rejection of Claims 14 and 25 and the present rejection is traversed at least for the reasons discussed above in connection with Claims 14 and 25.

15. In view of the above remarks, reconsideration of all claims in the application is requested. Respectfully submitted,

| | SIGNATURE | |
|---|---|---|
| **David E. Lovejoy** (US Reg. No.: 22,748) | *David E. Lovejoy* /david lovejoy/ | **Signature Date** 15 October 2006 |
| Mail Address | Customer No. | Communication |
| 102 Reed Ranch Rd. Tiburon, CA 94920-2025 USA | 21603 | Tel: (415) 435-8203 Fax: (415) 435-8857 e-mail:david.lovejoy@sbcglobal.net |